1. SOFTWARE ENGINEERING

Engineering: Engineering is the application of scientific, economic, social, and practical knowledge in order to invent, design, build, maintain, research, and improve structures, machines, devices, systems, materials and processes.

Software: Software is i) instructions(computer programs) that when executed provide desired function and performance. ii) data structures that enable the programs to adequately manipulate information and iii) documents that describe the operation and use of the programs.

Software is a general term for the various kinds of *program*s used to operate computers and related devices.

Software Engineering**:** *IEEE* defines software engineering as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

SOFTWARE CHARACTERISTICS

- **Software is developed or engineered, it is not manufactured**
  Unlike hardware, software is logical rather than physical. It has to be designed well before producing it. In spite of availability of many automated software development tools, it is the skill of the individual, creativity of the developers and proper management by the project manager that counts for a good software product.
- **Software does not "wear out"**
  As time progresses, the hardware components start deteriorating-they are subjected to environmental maladies such as dust, vibration, temperature etc. and at some point of time they tend to breakdown. The defected components can then be traced and replaced. But, software is not susceptible to the environmental changes. So, it does not wear out. The software works exactly the same way even after years it was first developed unless any changes are introduced to it. The changes in the software may occur due to the changes in the requirements. And these changes may introduce some defects in it thus, deteriorating the quality of software, so software need to maintain properly.
- **Most software is custom-built, rather than being assembled from existing components**
  Most of the engineered products are first designed before they are manufactured, Designing includes identifying various components for the product before they are actually assembled. Here several people can work independently on these components, thus making the manufacturing system highly flexible. In software, breading a program into modules is difficult task, since each module is highly interlinked with other modules. Further, it requires lot of skill to integrate different modules into one. Now a days the term component is widely used in software industry where object oriented system is in use.

CATEGORIES OF SOFTWARE

Software are mainly classified into two. **System software and Application software**.

System Software: System software is a set of programs which manages and controls the internal operations of a computer system. It is a collection of programs which is responsible for using computer resources efficiently and effectively. That is a collection of programs written to service other programs. The system software is again categorized as **Operating systems and Language processors.**

***Operating System (OS*)**: OS is considered as the fundamental system software. OS is an interface between the user and hardware devices. OS provides services like hardware control, memory management, multitasking etc.

Example: Windows 8, Windows 7, Windows XP, Ubuntu, Mac OS, Android, DOS, UNIX, Google OS etc.

Language Processors (LP): Computer can recognize only the machine language/ Binary language (0s and 1s). But programmers used to write programs in High Level Language (HLL) which is easily understandable to the humans. So there is a need of a translator which converts the programs written in HLL into an equivalent machine language. The system programs which performs this translation is called Language Processors. The different language processors are

i) **Assembler:** Coverts the program written in assembly language into its equivalent machine language.

ii) **Interpreter:** Converts the program written in High Level Language (HLL) into its equivalent machine language by converting and executing each line by line. If an error is there in the HLL code, it reports the error and execution terminates. Execution is resumed only after correcting the error in that line.

iii) **Compiler:** Compiler is similar to Interpreter which translates a High Level Language into its equivalent machine language. But Compiler compiles a whole program and then it lists out the errors along with its line numbers. If there are no errors in the HLL code, computer generates the object files. The process of translating a HLL into its equivalent machine language is termed as **compilation.**

Application Software: Application software are programs which are developed in order to perform a particular task or functionality. That is they are stand - alone programs that solve a specific business need. Application software are programs which are designed to run under an operating system. They range from word processors and Internet browsers to video games and media players. Application software is further classified into three categories.

i) Packages ii) Utilities iii) Customized Software

Packages: Application package software, or simply an application package, is a collection of software programs that have been developed for the purpose of being licensed to third-party organizations. Application packages are generally designed to support commonly performed business functions and appeal to multiple types of user organizations. Various packages are listed below.

- Word Processing Software: Allows users to create, edit a document.
  Example: MS Word, Word Pad, Note pad etc.
- Spreadsheet Software: Allows users to create document and perform calculation. Example: MS Excel, Lotus1-2-3 etc.
- Database Software: Allows users to store and retrieve vast amount of data. Example: MS Access, MySQL, Oracle etc.
- Presentation Graphic Software: Allows users to create visual presentation. Example: MS Power Point
- Multimedia Software: Allows users to create image, audio, video etc.
  Example: Real Player, Media Player etc.

Utilities: Utility software is a collection of one or more programs that helps the user in system maintenance tasks and in performing tasks of routine nature. Utility programs help the users in disk formatting, data compression, data backup, scanning for viruses etc. It improves the efficiency and performance of the computer.

Examples of utility software are:

- Anti-virus
- Registry cleaners
- Disk defragmenters
- Data backup utility
- Disk cleaner

Customized software: Custom software (also known as **bespoke software** or **tailor-made software**) is software that is specially developed for some specific organization or other user. Large companies commonly use custom software for critical functions, including content management, inventory management, customer management, human resource management etc.

Hierarchy from tokens to software:

Tokens → Instructions / LOC → Modules / Functions → Programs → Software → Packages

A package is a collection of various software. A software consists of various programs. A program may contain various functions or modules. Each module or function may have multiple instructions or Line Of Code (LOC). An instruction consist of various tokens.

2. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Life cycle model: A life cycle model describes the different activities that need to be performed to develop a software. It also shows the sequencing of these activities.

SDLC: A SDLC is a descriptive and diagrammatic representation of the software life cycle.

Need of SDLC:

- Encourages development of software in a systematic and disciplined manner.
- Provides precise understanding among team members, thus helps to avoid chaos during development phases.
- Reduces project failure
- Makes easier to allocate resources and persons during various phases of software development.

WATERFALL MODEL / CLASSIC LIFE CYCLE MODEL / LINEAR SEQUENTIAL MODEL

The first published model of software development process was derived from other engineering processes. The classical waterfall model is the most obvious way to develop software. But it is not a practical model because it cannot be used in actual software development. So Waterfall model is a theoretical way of developing software. Then why it is relevant because the various other life cycle model is based on this Waterfall model. Waterfall model is also called as **Linear sequential model or Classic life cycle model.** The various phases of Waterfall model is illustrated below.

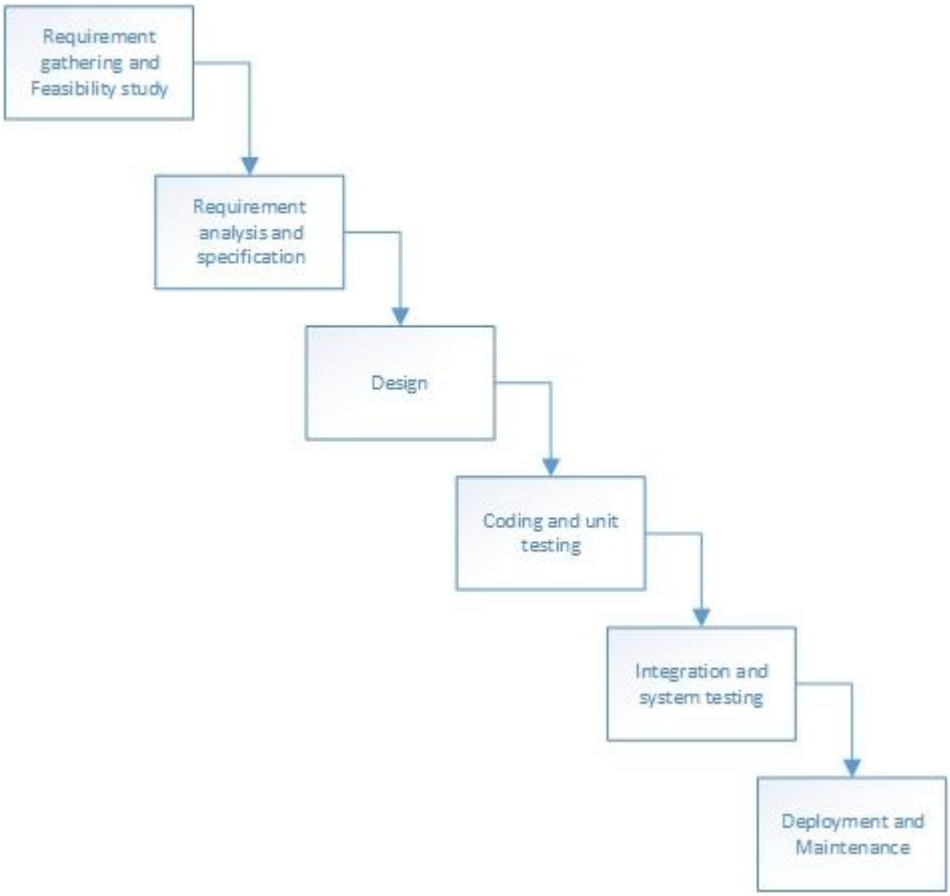


Figure 2.1: Waterfall model

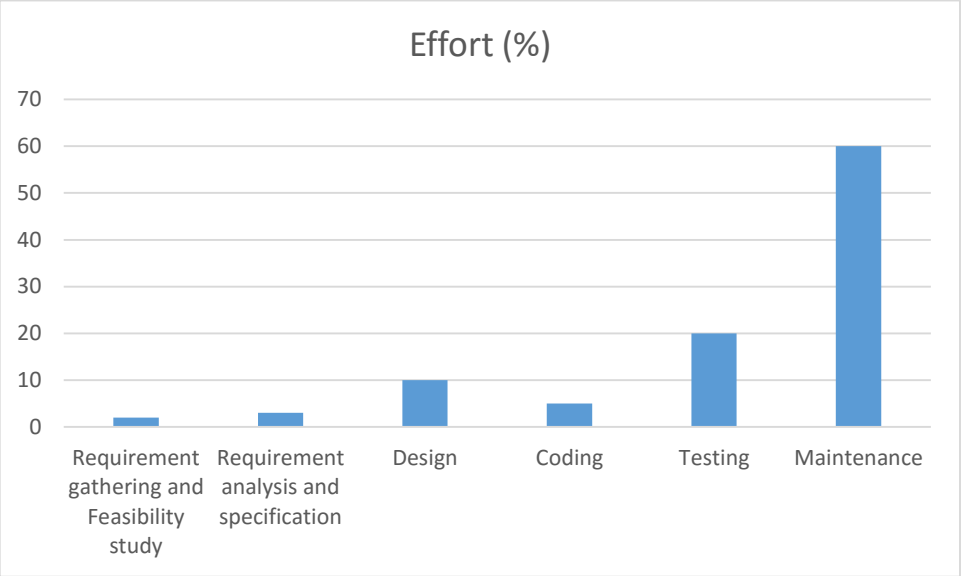
- The basic principle in this model is, dividing the entire life cycle into various phases. Each phase has a well -defined function or characteristics. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.
- No feed backing is provided between phases. That is, it is not possible to go back to a previous phase and make changes.
- **Requirement gathering:** This phase involves understanding what you need to design and what its intended purpose or function is. The requirements may be collected directly from the customer / client or by questionnaire survey or by visiting the target environment where the software is being deployed.
- **Feasibility study:** After identifying the various requirements feasibility study is conducted. The main aim of feasibility study is to determine whether it would be practically possible to develop the product. The identified requirements should be technically, economically and operationally feasible.
    1) *Technical feasibility*: Ensuring whether the requirements can be implemented using the existing technology or not.
    2) *Economic feasibility*: Ensuring whether the requirements can be implemented within the given budget or not.
    3) *Operational feasibility*: Focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture, and existing business processes.
- **Requirement analysis:** As per the requirements, the software and hardware needed for the proper completion of the project is analysed in this phase. Right from deciding which computer language should be used for designing the software, to the database system that can be used for the smooth

functioning of the software, such features are decided at this stage. The main activities in this phase are identifying inconsistent requirements and prioritizing the requirements.

- **Requirement specification:** During requirement specification, the requirements are documented in a suitable format. The document used for this purpose is called SRS (Software Requirement Specification). The components of SRS are
    1) *Functional requirements:* Specify the expected behaviour of the system - which outputs should be produced from the given inputs
    2) *Non – functional requirements:* Specifies the performance constraints on the software system.
    3) *Goal of implementation:* Specifies about the need of implementing the software and mentioning its application areas.
- **Design:** The goal of design phase is to transform the requirements mentioned in the SRS document into a structure that is suitable for implementation in some programming languages. The algorithm (pseudo-code) of the program or the software code to be written in the next stage, is created now. This algorithm forms the backbone for the actual coding process. Proper planning relating to the design of user interface, flowcharts is done here. Design process is also documented to form the *design document* of the software. The activities carried out during design phase are
    i) Developing algorithms to implement the various requirements / functionalities.
    ii) Developing flow charts, DFDs, Structure charts, UML diagrams etc.
    iii) Choosing the appropriate data structures.
    iv) Choosing the suitable design methodology with architectural design and detailed design.
- **Coding:** Based on the algorithm or flowchart designed, the actual coding of the software is carried out at this stage. The flowcharts / algorithms are converted into instructions written in a programming language.
- **Testing:** The software designed, needs to go through constant software testing and error correction processes to find out if there are any flaw or errors. Testing is done so that the client does not face any problem during the installation of the software. Testing is done with the intention to find errors. Testing is also documented to form *Test reports*.
- **Unit testing:** In modular programming methodology, the software is divided into various modules or units. A unit is the smallest testable part of an application like functions, classes, procedures, interfaces. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use. Usually performed by developers itself.
- **Integration and System testing:** The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. System testing usually consist of three different testing methodologies.
    i) Alpha testing: Testing performed by the developing team itself.
    ii) Beta testing: Testing performed by a friendly set of customers.
    iii) Acceptance testing: Testing performed by the customer after the product delivery to determine whether to accept or reject the delivered product.
- **Deployment:** It mainly refers the product release, installation of product in the target environment, activating the product, Updating etc.
- **Maintenance:** Modification of a software product after delivery to correct faults, to improve performance or other attributes. Maintenance activities are classified as follows:
    i) *Corrective maintenance*: Correcting the errors that are not identified during the product development phase.

ii) *Perfective maintenance*: Improving the implementation of the system and enhancing the system functionalities according to the client's requirements.
iii) *Adaptive maintenance*: Usually done for porting the software to work in a new environment (For example, new Operating system or new hardware platform etc.)
iv) *Preventive maintenance*: Implementing changes to prevent the occurrence of errors.

- **Advantages / Pros of Waterfall model**
  i) Relatively simple to understand
  ii) Easier to allocate resources and persons.
  iii) Each phase of development proceeds sequentially.
- **Disadvantages / Cons of Waterfall model**
  i) Requirements need to be specified before the development proceeds.
  ii) No feedback between phases. Hence not applicable in projects where requirements are changing rapidly.
  iii) Do not involve risk management.
  iv) No scope for error correction
  v) Highly impractical for most projects because only few projects can be divided into such water-tight phases.
- **Applications**
  It can be applied in long term projects or the projects where requirements are finite.
- **Effort Distribution among different phases**



ITERATIVE WATERFALL MODEL

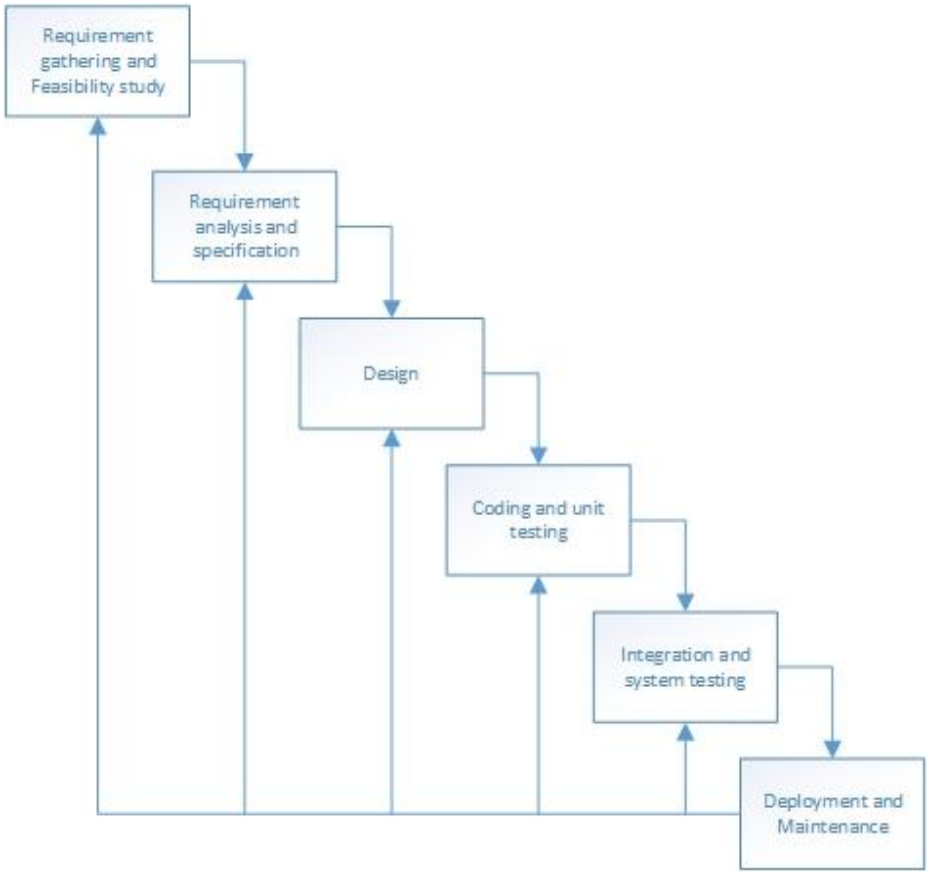


Figure: Iterative Waterfall model

Classical Waterfall model is an idealistic model. So we made necessary changes to the classical waterfall model, so that it becomes applicable to practical software development projects. Essentially the main change in Iterative waterfall model is in the form of providing feedback paths from every phase to its preceding phases as shown in the above figure. The feedback paths allow for correction of the errors committed during a phase, as and when these are detected in the later phase.

DISADVANTAGES /CONS:

- Cannot handle different types of risks evolved during development.
- Difficult to follow a rigid phase sequence. *That is once a team member complete his work earlier, he would have to be idle for other members to complete their work, before proceeding into his new activity.*

PROTOTYPING MODEL

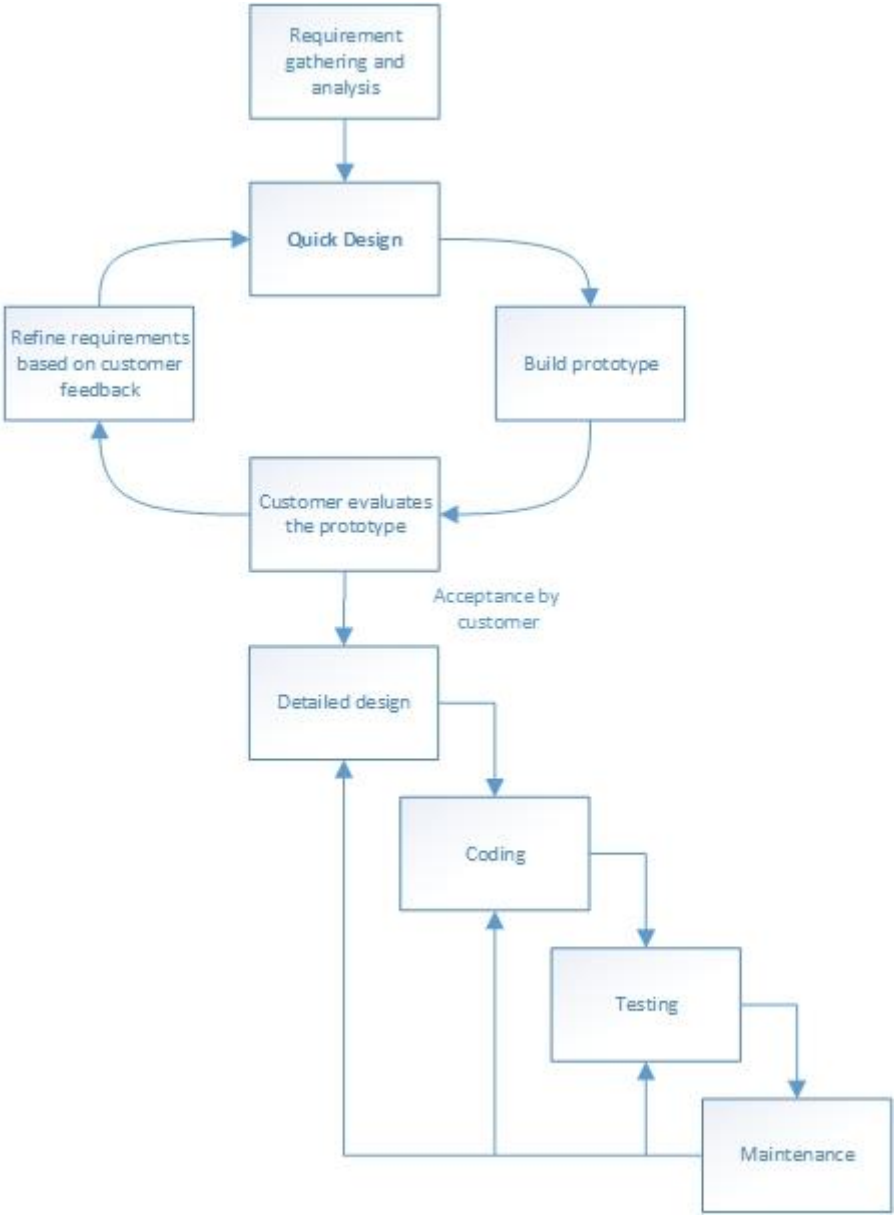


Figure: Prototyping model

- Prototyping model requires that before carrying out the development of the actual software, a working prototype of the system should be built. A prototype is a toy implementation or dummy model of the actual system. It is built using several shortcuts like dummy functions, GUI outlook using buttons etc.
- A prototype model has limited functional capabilities, low reliability and inefficient performance as compared to the actual software. The prototype gives the user an actual feel of the system.
- In this model it is assumed that all the requirements may not be known at the start of the development of the system.
- This model allows the users to interact and experiment with a working model of the system known as prototype.
- The code for the prototype is usually thrown away.
- Prototype can be created by the following approaches:
  i) By creating the main user interfaces without any substantial coding so that users can get a feel of how the actual system will appear.

ii) By abbreviating a version of the system that will perform limited subsets of functions.

iii) *By using system components to illustrate the functions that will be included in the system to be developed.*

- **Execution of prototyping model:** Prototyping begins with an initial requirement gathering phase. A quick design is carried out and a prototype is built. The developed prototype is submitted to the customer for his evaluation. Based on the customer feedback, the requirements are refined and the prototype is suitably modified. This cycle of obtaining customer feedback and modifying the prototype continues till the customer approves the prototype. Once the customer approves the prototype, the actual system is developed using the iterative waterfall approach.

PHASES IN PROTYPING MODEL

- **Requirements gathering and analysis:** This phase involves understanding what you need to design and what its intended purpose or function is. The requirements may be collected directly from the customer / client or by questionnaire survey or by visiting the target environment where the software is being deployed. The main activities in requirement analysis phase are identifying inconsistent requirements and prioritizing those requirements. After that a suitable SRS document is prepared.
- **Quick design:** When requirements are known, a preliminary design or quick design for the system is created. It is not a detailed design and includes only the important aspects of the system, which gives an idea of the system to the user. A quick design helps in developing a prototype.
- **Build prototype:** Information gathered from quick design is modified to form the first prototype, which represents the working model of the actual system. It is a dummy model or a toy implementation of the actual software. It provides the look and feel of the actual system with limited functionalities.
- **Customer evaluation / User evaluation:** The prototype is presented before the user for proper evaluation in order to recognize the strengths and weaknesses such as what is to be added or removed. Comments and suggestions are collected from the users and are provided to the developer.
- **Refining prototype:** Based on the customer suggestions, the current prototype is refined. That is a new prototype is developed, incorporating the customer suggestions. The new prototype is presented to the user for evaluation. Customer evaluated it and provide modifications. Based on that, again the model is refined. This process continues until customer satisfied with the developed prototype. A final system is developed on the basis of the final prototype.
- **Detailed Design:** The goal of this phase is to convert the prototype model into target system with a detailed design. The activities carried out during this phase are
    i) Developing algorithms to implement the various requirements / functionalities.
    ii) Choosing the appropriate data structures.
- **Coding:** Based on the algorithm or flowchart designed, the actual coding of the software is carried out at this stage. The flowcharts / algorithms are converted into instructions written in a programming language.
- **Testing:** The software designed, needs to go through constant software testing and error correction processes to find out if there are any flaw or errors. Testing is done so that the client does not face any problem during the installation of the software. Testing is done with the intention to find errors. Testing is also documented to form *Test reports*.
- **Unit testing:** In modular programming methodology, the software is divided into various modules or units. A unit is the smallest testable part of an application like functions, classes, procedures, interfaces. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use. Usually performed by developers itself.

- **Integration and System testing:** The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. System testing usually consist of three different testing methodologies.
    i) Alpha testing: Testing performed by the developing team itself.
    ii) Beta testing: Testing performed by a friendly set of customers.
    iii) Acceptance testing: Testing performed by the customer after the product delivery to determine whether to accept or reject the delivered product.
- **Maintenance:** Modification of a software product after delivery to correct faults, to improve performance or other attributes. Maintenance activities are classified as follows:
    i) *Corrective maintenance*: Correcting the errors that are not identified during the product development phase.
    ii) *Perfective maintenance*: Improving the implementation of the system and enhancing the system functionalities according to the client's requirements.
    iii) *Adaptive maintenance*: Usually done for porting the software to work in a new environment (For example, new Operating system or new hardware platform etc.)
    iv) *Preventive maintenance*: Implementing changes to prevent the occurrence of errors.
- **ADVANTAGES / PROS:**
    - ✓ Improved customer satisfaction.
    - ✓ Developer gains experience from developing prototypes which results in the better implementation of requirements.
    - ✓ Chance of occurring errors is comparatively less.
- **DISADVANTAGES /CONS:**
    - ✓ Model is time consuming and expensive.
    - ✓ Developer may compromise with the quality of the software
    - ✓ Prototyping can lead to false expectations. For example, a situation may be created where the user believes that the development of the system is finished when it is not.
- **APPLICATIONS**
    - ✓ Used in the development of systems where GUI has much importance.
    - ✓ Prototype model should be used when the desired system needs to have a lot of interaction with the end users. Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model.
    - ✓ Excellent for designing good human computer interface systems.
    - ✓ Can be used when the technical solution is not clear for the developer.

SPIRAL MODEL

- Spiral model was proposed Barry Boehm in 1986. This model combines the best features of the prototyping model and waterfall model and is advantageous for large, complex and expensive projects.
- IEEE defines the Spiral model as '*a model of the software development process in which the constituent activities, typical requirements analysis, preliminary and detailed design, coding, integration and testing are performed iteratively until the software is complete*'.
- The objective of Spiral model is to emphasize management to evaluate and resolve risks in the software projects.
- Different risks in the software project are project overruns, changed requirements, loss of key project personnel, delay of necessary hardware, competition with other software developers and technological breakthroughs etc.
- Rather than represent the software process as a sequence of activities with some backtracking from one activity to another, the process is represented as a spiral. Each loop in the spiral represents a phase

of the software process. Thus the inner most loop might be concerned with system feasibility, the next loop with system requirements definition, the next loop with system design and so on.

- Each loop in the Spiral model is divided into four quadrants. Each quadrant has a well-defined set of activities. The four quadrants are:
  i) Determine objectives, alternatives and constraints **(Objective setting)**
  ii) Evaluate alternatives, identify and resolve risks. **(Risk assessment and reduction)**
  iii) Develop and verify next – level product **(Development and validation)**
  iv) Plan the next phase. **(Planning)**

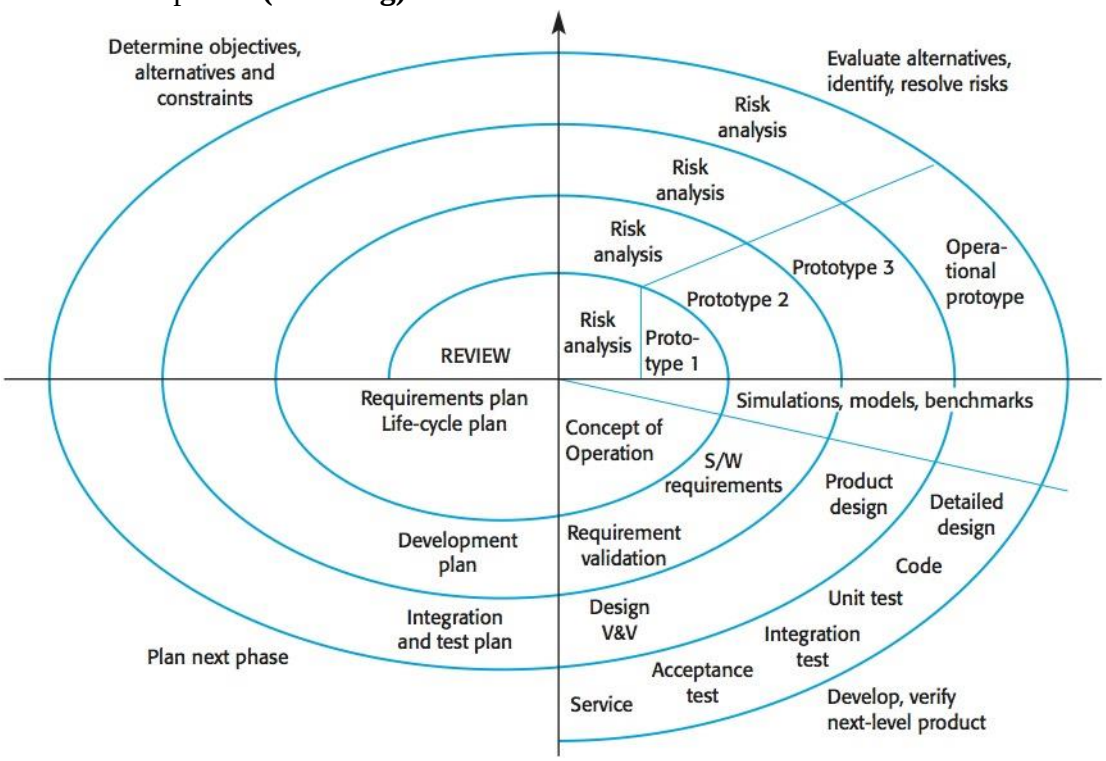


Figure: Spiral model

PHASES OR QUADRANTS IN SPIRAL MODEL

- **Determine objectives, alternatives and constraints:** Specific objectives for that phase of the project are defined. Constraints on the process and the product are identified and a detailed management plan is drawn up. Project risks are identified. Alternative strategies, depending on these risks, may be planned.
- **Evaluate alternatives, identify and resolve risks**: For each of the identified risks, a detailed analysis is carried out. Steps are taken to reduce the risk. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.
- **Develop and verify next – level product:** After risk evaluation, a development model for the system is then chosen. Risk management considers the time and effort to be devoted to each project activity such as planning, configuration management, quality assurance, verification and testing.
- **Plan the next phase:** The project is reviewed and a decision made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.
- One of the key features of Spiral model is that each cycle is completed by a review conducted by the individuals or users. This includes review of all the intermediate products, which are developed during the cycles.

- The radius of the spiral represents the cost of development, whereas the angular displacement represents the progress in development.
- The important distinction between Spiral model and other software process models is the explicit consideration of risk in the spiral model. Informally, Risk is something which can go wrong.
- There are no fixed phases such as specification or design in the spiral model. The spiral model encompasses other process models. Prototyping may be used in one spiral to resolve requirements uncertainties and hence reduce risk. This may be followed by a conventional waterfall development.
- **ADVANTAGES / PROS**
  - ✓ High amount of risk analysis hence, avoidance of Risk is enhanced.
  - ✓ Good for large and mission-critical projects.
  - ✓ Strong approval and documentation control.
  - ✓ Additional Functionality can be added at a later date.
  - ✓ Software is produced early in the software life cycle.
- **DISADVANTAGES / CONS**
  - ✓ Can be a costly model to use.
  - ✓ Risk analysis requires highly specific expertise.
  - ✓ Project's success is highly dependent on the risk analysis phase.
  - ✓ Doesn't work well for smaller projects.
- **APPLICATIONS**
  - ✓ When costs and risk evaluation is important
  - ✓ For medium to high-risk projects
  - ✓ Long-term project commitment unwise because of potential changes to economic priorities
  - ✓ Projects in which the users are unsure of their needs
  - ✓ Projects in which the requirements are complex.

SELECTING AN APPROPRIATE LIFE CYCLE MODEL FOR A PROJECT

- ❖ **Characteristics of the software to be developed:** If the software is simple data processing application an iterative waterfall model should be sufficient. An evolutionary model is a suitable model for object oriented development projects.
- ❖ **Characteristics of the development team:** If the development team is experienced in developing similar projects, then we can follow iterative waterfall model. If the team is entirely novice, then a prototyping model can be used.
- ❖ **Characteristics of the customer:** If the customer is not familiar with computers, then requirements are likely to change frequently. So prototyping model may be necessary to reduce the later change requests from customers.